AMENDMENTS TO THE CLAIMS:

1.(currently amended): An information processing system including at least one terminal apparatus and at least one program execution apparatus, wherein

each of said at least one terminal apparatus includes a message transmitting unit which transmits a message containing version information indicating a program version; [[and]]

wherein the version information is defined in different namespaces, when written in programming language C++, for preventing a collision of names of functions and variables in different versions of the program; and

each of said at least one program execution apparatus includes,

a message receiving unit which receives said message containing version information indicating a program version, from one of said at least one terminal apparatus,

a program storing unit which stores one or more program components,

a pre-transfer information management table which holds information on said one or more program components stored in said program storing unit,

a program memory unit which is allocated to an activated process, and temporarily stores at least one program component transferred from said program storing unit,

a post-transfer information management table which holds information on said at least one program component stored in said program memory unit, and

a program executing unit which dynamically links one of said one or more program components corresponding to said version information contained in said message received by said message receiving unit, to said program memory unit, so as to enable execution of said one of said one or more program components in said process, and which changes, without stopping transaction processing, a plurality of versions of a plurality of software components

which are distributed over a plurality of the terminal apparatuses when more than one version of a software component may coexist in each of the terminal apparatus.

2. (original): An information processing system according to claim 1, wherein said post-transfer information management table indicates for each of said at least one program component a reference count which indicates the number of executions in which said each of said at least one program component are currently referred to.

3. (original): An information processing system according to claim 2, wherein said program executing unit removes one of said at least one program component from said program memory unit when said reference count for said one of said at least one program component is zero, and information on a newer version of said one of said at least one program component is included in the said post-transfer information management table.

4. (original): An information processing system according to claim 1, wherein said message received by said message receiving unit further contains an evaluation flag which indicates whether or not a program component is under evaluation.

5. (original): An information processing system according to claim 4, wherein said program executing unit executes said program component for which said evaluation flag is contained in said message, in a separate process, when the evaluation flag indicates that said program component for which said evaluation flag is contained in said message is under evaluation.

6. (original): An information processing system according to claim 1, wherein said post-transfer information management table indicates for each of said at least one program component an evaluation count which indicates the number of evaluations which are currently performed on said each of said at least one program component.

7. (original): An information processing system according to claim 6, wherein said program executing unit terminates execution of one of said at least one program component in a separate process when said evaluation count corresponding to said one of said at least one program component is zero.

8. (original): An information processing system according to claim 1, further comprising a management apparatus which sets and manages said one or more program components stored in said program storing unit and said information on said one or more program components held in said pre-transfer information management table.

9. (cancelled)

10. (currently amended): A program execution apparatus comprising:

a message receiving unit which receives a message containing version information indicating a program version;

wherein the version information is defined in different namespaces, when written in programming language C++, for preventing a collision of names of functions and variables in different versions of the program;

a program storing unit which stores one or more program components;

a pre-transfer information management table which holds information on said one or more program components stored in said program storing unit;

a program memory unit which is allocated to an activated process, and temporarily stores at least one program component transferred from said program storing unit;

a post-transfer information management table which holds information on said at least one program component stored in said program memory unit; and

a program executing unit which dynamically links one of said one or more program components corresponding to said version information contained in said message received by said message receiving unit, to said program memory unit, so as to enable execution of said one of said one or more program components, and which changes, without stopping transaction processing, a plurality of versions of a plurality of software components which are distributed over a plurality of computers when more than one version of a software component may coexist in each of the computer.

11. (currently amended): A distributed processing system including at least one client apparatus and a plurality of server apparatuses, and realizing an N-tier client-server environment, wherein

each of said at least one client apparatus includes a client stub processing unit which executes stub processing of a first message which contains version information indicating a program version; [[and]]

wherein the version information is defined in different namespaces, when written in programming language C++, for preventing a collision of names of functions and variables in different versions of the program; and

each of said plurality of server apparatuses includes,

a server skeleton processing unit which executes skeleton processing of said first message containing version information indicating a program version,

a distributed-object storing repository which stores one or more distributed objects,

a pre-transfer information management table which holds information on said one or more distributed objects stored in said distributed-object storing repository,

a memory which is allocated to an activated process, and temporarily stores at least one distributed object transferred from said distributed-object storing repository,

a post-transfer information management table which holds information on said at least one distributed object stored in said memory,

a distributed object execution control unit which dynamically links one of said one or more distributed objects corresponding to said version information contained in said first message of which skeleton processing is executed by said server skeleton processing unit, to said memory, so as to enable execution of at least one function in said one of said one or more distributed objects, and which changes, without stopping transaction processing, a plurality of versions of a plurality of software components which are distributed over a plurality of the client

apparatuses when more than one version of a software component may coexist in each of the client apparatus, and

a server stub processing unit which executes stub processing of a second message containing said version information so as to transmit the second message to another of said plurality of server apparatuses.

12. (original): A distributed processing system according to claim 11, wherein said post-transfer information management table indicates for each of said at least one distributed object a reference count which indicates the number of executions in which said each of said at least one distributed object are currently referred to.

13. (original): A distributed processing system according to claim 12, wherein said distributed object execution control unit removes one of said at least one distributed object from said memory when said reference count for said one of said at least one distributed object is zero, and information on a newer version of said one of said at least one distributed object is included in the said post-transfer information management table.

14. (original): A distributed processing system according to claim 11, wherein each of said first and second messages further contains an evaluation flag which indicates whether or not a distributed object is under evaluation.

15. (original): A distributed processing system according to claim 14, wherein said distributed object execution control unit executes said distributed object for which said

evaluation flag is contained in said first message, in a separate process, when the evaluation flag indicates that said program component for which said evaluation flag is contained in said first message is under evaluation.

16. (original): A distributed processing system according to claim 11, wherein said post-transfer information management table indicates for each of said at least one distributed object an evaluation count which indicates the number of evaluations which are currently performed on said each of said at least one distributed object.

17. (original): A distributed processing system according to claim 16, wherein said distributed object execution control unit terminates execution of one of said at least one distributed object in a separate process when said evaluation count corresponding to said one of said at least one distributed object is zero.

18. (original): A distributed processing system according to claim 11, further comprising a management server which sets and manages said one or more distributed objects stored in said distributed-object storing repository and said information on said one or more distributed objects held in said pre-transfer information management table.

19. (cancelled)

20. (currently amended): A server apparatus comprising:

a server skeleton processing unit which executes skeleton processing of a first message containing version information indicating a program version;

wherein the version information is defined in different namespaces, when written in programming language C++, for preventing a collision of names of functions and variables in different versions of the program;

a distributed-object storing repository which stores one or more distributed objects;

a pre-transfer information management table which holds information on said one or more distributed objects stored in said distributed-object storing repository;

a memory which is allocated to an activated process, and temporarily stores at least one distributed object transferred from said distributed-object storing repository;

a post-transfer information management table which holds information on said at least one distributed object stored in said memory;

a distributed object execution control unit which dynamically links one of said one or more distributed objects corresponding to said version information contained in said first message of which skeleton processing is executed by said server skeleton processing unit, to said memory, so as to enable execution of at least one function in said one of said one or more distributed objects, and which changes, without stopping transaction processing, a plurality of versions of a plurality of software components which are distributed over a plurality of the client apparatuses when more than one version of a software component may coexist in each of the client apparatus; and

a server stub processing unit which executes stub processing of a second message containing said version information so as to transmit the second message to another server apparatus.

21. (currently amended): A method for updating a program component loaded in a server in an N-tier client-server environment, comprising the steps of:

(a) storing in said server one or more program components;

(b) holding information on said one or more program components stored in said server, in a pre-transfer information management table;

(c) transmitting a first message containing version information indicating a program version, from a client to said server;

wherein the version information is defined in different namespaces, when written in programming language C++, for preventing a collision of names of functions and variables in different versions of the program;

(d) receiving said first message by said server;

(e) dynamically linking one of said one or more program components corresponding to said version information contained in said first message, to a memory which is allocated to an activated process in said server, so as to enable execution of said one of said one or more program components in said process;

(f) changing, without stopping transaction processing, a plurality of versions of a plurality of software components which are distributed over a plurality of the clients when more than one version of a software component may coexist in each of the clients ~~client~~;

(g) holding in a post-transfer information management table information on at least one program component stored in said memory; and

(h) transmitting to another server a second message containing said version information.

22. (original): A method according to claim 21, wherein said post-transfer information management table indicates for each of said at least one program component a reference count which indicates the number of executions in which said each of said at least one program component are currently referred to.

23.(previously presented): A method according to claim 22, further comprising the step of,

(i) removing one of said at least one program component from said memory when said reference count for said one of said at least one program component is zero, and information on a newer version of said one of said at least one program component is included in the said post-transfer information management table.

24. (original): A method according to claim 21, wherein each of said first and second messages further contains an evaluation flag which indicates whether or not a program component is under evaluation.

25. (original): A method according to claim 24, wherein, in said step (f), said program component for which said evaluation flag is contained in said first message is executed in a

separate process when the evaluation flag indicates that said program component for which said evaluation flag is contained in said first message is under evaluation.

26. (original): A method according to claim 21, wherein said post-transfer information management table indicates for each of said at least one program component an evaluation count which indicates the number of evaluations which are currently performed on said each of said at least one program component.

27.(previously presented): A method according to claim 26, further comprising the step of,

(j) terminating execution of one of said at least one program component in a separate process when said evaluation count corresponding to said one of said at least one program component is zero.

28.(previously presented): A method according to claim 21, further comprising the step of,

(k) setting and managing said one or more program components stored in said server and said information on said one or more program components held in said pre-transfer information management table, by using a management apparatus.

Claims 29-31.(cancelled)